PATENT
Attorney Docket No.: 40101-10701

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	)	
	)	
Maarten A. KONING	)	
	)	
Serial No.: 09/522,541	)	Group Art Unit: 2127
	)	
Filed: March 10, 2000	)	Examiner: M. Banankhah
	)	
For: TASK CONTROL BLOCK FOR A COMPUTING ENVIRONMENT	)	**Board of Patent Appeals and Interferences**

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

APPEAL BRIEF UNDER 37 C.F.R. § 1.192

In support of the Notice of Appeal filed July 7, 2004, and pursuant to 37 C.F.R. § 1.192, appellants present in triplicate their appeal brief in the above-captioned application.

This is an appeal to the Board of Patent Appeals and Interferences from the Examiner's final rejection of claims 1-21 in the final Office Action dated January 14, 2004. The appealed claims are set forth in the attached Appendix A.

1. Real Party in Interest

This application is assigned to Wind River Systems, Inc., the real party in interest.

2. Related Appeals and Interferences

There are no other appeals or interferences which would directly affect, be directly affected, or have a bearing on the instant appeal.

3. Status of the Claims

Claims 1-21 are pending.

Claims 1-21 have been rejected in the final Office Action and are involved in this appeal.

4. Status of Amendments

Appellants have not submitted any amendments to the claims as filed in the original application.

5. Summary of the Invention

The present invention is directed to an improved task control block. A first portion of the task control block is implemented in system memory space allocated to the operating system (the system task control block) and a second portion of the task control block is implemented in user memory space allocated to the user task (the user task control block). *Specification*, page 5, lines 9-14. Read/write operations on the data structures in the system task control block may be performed by the operating system

tasks, but are not directly accessible to tasks executing outside system memory space. The data structures in the user control task block may be directly read/written by the corresponding task. *Id.* at p.5, lines 14-20.

System task control blocks are located within the system space and user task control blocks and task stacks are located within user space. Thus, a user task may directly access the user task control block and the task stack in its memory location, but may not directly access the system task control block in the system memory location. The system space may further contain a current task data structure for storing task state information and a context switch routine for performing context switching. *Id.* at p. 7, lines 11-24.

The task information stored in the system task control block may be used during context switching to save and restore the state of the computing environment for a task. The state of the computing environment for a currently executing task is stored in a current task data structure. As part of the context switch between tasks, the contents of the current task data structure are saved into the system task control block and the contents of the system task control block for the newly executing task are loaded into the current task data structure. *Id.* at p. 8, lines 12-24.

The user task control block stores additional task information data structures for the task. The task may directly access the information in the user task control block without a need for a system call. *Id.* at p. 8, line 26 - p. 9, line 13.

6. Issues

I. Whether claims 1-21 are anticipated under 35 U.S.C. § 102(b) by Mateosian, "Operating System Support - The Z8000 Way," *Computer Design*, May 1982 (hereinafter "*Mateosian*").

7. Grouping of Claims

Claims 1-13 may stand or fall together. Claims 14-16 may stand or fall together. Claims 17-21 may stand or fall together.

8. Argument

I. The Rejection of Claims 1-21 Under 35 U.S.C. §102(b) as Being Anticipated by Mateosian, "Operating System Support - The Z8000 Way," *Computer Design*, May 1982 Should Be Reversed

A. The Examiner's Rejection

In the final office action the Examiner asserted that *Mateosian* describes the system and method for a task control block for a computing environment. Appellants disagree with the rejection of claims 1-21 under 35 U.S.C. §102(b). For at least the following reasons, the Board should reverse the rejection.

The Examiner asserted in the Final rejection that "[a]pplicants have not submitted sufficient evidence to rebut the strong prima facie case of obviousness established by the Examiner." (Final Office Action, p. 7). However, the Examiner's

rejections were base on 35 U.S.C. §102(b), which bars inventions from patent where:

> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of the application for patent in the United States.

The appropriate standard for a rejection under §102 is that the claim is anticipated by the cited reference, not obviousness. (*See* MPEP §706.02, IV). Non-obviousness is not a requirement under §102. (*See id.*). Therefore, the Examiner's statement regarding obviousness is irrelevant to the §102 rejection and the appellants do not bear any burden of rebutting the Examiner's conclusion of obviousness.

B. The Cited Reference Does Not Disclose a Task Control Block Including a Number of Task Information Data Structures that Contain Task Information, as Recited in Claim 1.

The Examiner asserted in the Final rejection that Figure 3 of *Mateosian* discloses a system including: (1) a task; (2) a first task control block associated with the task and located in a first area of the memory space, the first task control block including a number of first task information data structures that contain first task information; (3) a second task control block associated with the task and located in a second area of the memory space, the second task control block including a number of second task information data structures that contain second task information. (*See* Final Office Action, pp.3-4). Figure 3 of *Mateosian* merely lists "software components [which] allow manipulation of hardware and applications software, and represent system services that

all operating systems must apply." (*Mateosian*, p.256, left col., lines 36-38). The diagram presented in figure 3 groups software components into broad categories based on function. (i.e., Process Manager, Event Queue / Semaphore Manager, Memory Allocator, etc.). (*See id.*, Fig. 3). Each category may contain one or more components and may illustrate the functional purpose for each category of components. (*See id.*, Fig. 3). The categories are merely illustrations for visual understanding and do not provide any structural limitations. In particular, *Mateosian* is silent on data structure and memory storage allocations.

The Examiner stated that "the task control block located in memory stack and including a number of task information data structure is taught by *Mateosian* in Fig. 3, under process manager." (Final Office Action, p.5) (citation omitted). The Examiner equates the illustration of the Process Manager category with the task control block of the present invention. The Examiner further equates the software components of *Mateosian* with the task information data structures of the present invention. (Final Office Action, p. 5). In *Mateosian*, the software components listed as part of the Process Manager category are required functions of an operating system. However, they are not task information data structures. Figure 3 of *Mateosian* illustrates the functional components an operating system must perform, but does not indicate the data or structural implementations for these functional components. On the contrary, the present invention requires task information to be stored within at least one task information data structure of a memory

space for each of the two task control blocks.

It is well known in the computer science art that a control block is "[a] storage area containing (in condensed, formalized form) the information required for the control of a task, function, operation, or quantity of information." (McGraw-Hill Dictionary of Scientific and Technical Terms, 4th ed., p.425, attached hereto as Appendix B.). Thus, it follows that one skilled in the art would understand that the task control block of the present invention is a storage area which contains information required for the control of a task. As discussed above, *Mateosian* is silent on the data and structural implementation of the software components listed in Figure 3. *Mateosian* did not relate the functions and categories listed in Figure 3 to memory / storage areas or data structures. Moreover, *Mateosian* does not indicate a designated, condensed and formalized storage area solely dedicated for the use of the software components under the Process Manager category. Thus, *Mateosian* could not have contemplate the task control block as recite in claim 1 of the present invention.

Accordingly, appellants submit that Mateosian does not disclose a system including a first task control block associated with the task and located in a first area of the memory space, the first task control block including a number of first task information data structures that contain first task information and a second task control block associated with the task and located in a second area of the memory space, the second task control block including a number of second task information data structures that

contain second task information.

C. The Cited Reference Does Not Disclose Two Task Control Blocks Located in Two Separate Areas of a Memory Space, Wherein the First Area of the Memory Space is Not Directly Accessible by the Task and the Second Area of the Memory Space is Directly Accessible by the Task, as Recited in Claim 1.

The Examiner asserts that in the Final rejection that *Mateosian* describes a system including a first task control block associated with the task and located in a first area of the memory space; a second task control block associated with the task and located in a second area of the memory space; wherein the first area of the memory space is not directly accessible by the task, and the second area of the memory space is directly accessible by the task. As discussed above, Fig. 3 of *Mateosian* illustrates various software components required of an operating system and their functional categories. However, these categories were not related to memory / storage areas nor were they related to data structures. Furthermore, *Mateosian* does not disclose, nor does it suggest that the software components are implemented in two different sets of data structures, where each set is stored in a separate memory space. Instead, *Mateosian* discloses to the contrary, allowing every software component disclosed in Figure 3 to run in system mode. (*See Mateosian*, p. 256, right col., lines 55-56). Thus, any two categories of software components will have access to the same memory space.

The Examiner asserts that "[t]he program status area pointer control registers and the system mode stack register are all inaccessible from normal mode, and normal mode stack register is accessible from system mode" is the equivalent as the two memory spaces recited by claim 1 of the present invention where the first area of the memory space is not directly accessible by the task, and the second area of the memory space is directly accessible by the task. (Final Office Action, p. 3) (citations omitted). *Mateosian* discloses two different modes of the operating system: (1) a normal mode and (2) a system mode. (*See Mateosian*, p. 256, right col., lines 16-20). Under the normal mode, only the normal mode stack register is accessible, while under the system mode, the normal mode stack register in addition to the refresh register, the program status area pointer (PSAP) control registers, and the system mode stack register are all accessible. (*See id.*, p. 256, right col., lines 21-31). Nevertheless, the normal mode register and the registers accessible only via system mode disclosed in *Mateosian* are not equivalent to the areas of memory space recited in claim 1 of the present invention. Although *Mateosian* discloses registers accessible only in system mode and a register that is accessible in both normal and system modes, *Mateosian* does not disclose or even suggest that these registers are the same areas of memory space which contain task control blocks. The present invention limits both the first and second areas of memory space to include a first and second task control block, respectively.

The Examiner states that "[s]ections of the first task control block and of the second control block, which are different data structures depicted in Fig. 3, are on different part of the memory." As discussed above, appellants respectfully submit that the categories illustrated in Figure 3 are not task control blocks. Furthermore, *Mateosian* does not disclose data structures, nor does it contemplate implementing each category of software components in a set of data structures. In addition, even if it were considered that Figure 3 depicts task control blocks including a number of task information data structure, which the appellants submit that it does not, *Mateosian* does not locate the alleged "task control blocks" within separate memory spaces where one memory space is not directly accessible by the task while the other memory space is directly accessible by the task.

Mateosian permits all the software components listed in Figure 3 to run under system mode, where all registers are accessible. Therefore, the software components of Figure 3 may utilize any of the registers. The Examiner states that the box at the top left corner of Figure 3 (the "Process Manager" category) and the middle box in the top row of Figure 3 (the "Event Queue / Semaphore Manager" category) are on different parts of the memory. However, because the "Processor Manager" software components and the "Event Queue / Semaphore Manager" software component may both run in system mode, under the descriptions provided by *Mateosian*, component may utilize any of the following registers: the normal mode stack register, the refresh register,

the program status area pointer (PSAP) control registers, and the system mode stack register. It is also possible that all of the software components in both categories are all within one register, such as the system mode stack register. Therefore, the Examiner's exemplary "task control blocks" do not illustrate a first and second areas of memory space in which a first and second task control block are respectively located, such that the first area of memory space is not directly accessible by the task whereas the second area memory space is directly accessible by the task.

In light of the above, appellants submit that *Mateosian* does not disclose two areas of memory space each including a task control block wherein the first area of the memory space is not directly accessible by the task, and the second area of the memory space is directly accessible by the task. Therefore, for all of the reasons discussed in Sections 8.I.B and 8.I.C, appellants respectfully request that the Board overturn the Examiner's rejection of claim 1 and all the rejected claims dependent therefrom (claims 2-13).

D. The Cited Reference Does Not Disclose Each and Every Element of the <u>Present Invention as Recited in Claim 14</u>

The Examiner rejected claim 14 using the same arguments as those asserted against claim 1 to reject claim 14 under 35 U.S.C. §102(b) as being anticipated by *Mateosian*. (*See* Final Office Action, p. 2-3). However, in order to render a claim anticipated under §102, a single prior art reference must disclose each and every element

of the claim in exactly the same way as recited in the claim. *(See Lindeman Machinenfabrik v. Am Hoist and Derrick*, 730 F.2d 1452, 1458 (Fed. Cir. 1984)). If any claimed element is absent from the prior art reference, there is no anticipation. *(See Rowe v. Dror*, 112 F.3d 473, 478 (Fed. Cir. 1997)). The Examiner did not meet the requirements for a §102 rejection. The Examiner failed to address the limitation "loading an address for the location of the second task control block into the first task control block." Nowhere in the First and Final rejections, did the Examiner discuss an address for the location of the second task control or the step of loading it into the first task control block. Appellants submit that the Examiner's §102 rejection of claim 14 was improper because he had failed to demonstrate that the prior art discloses each and every element of the claim.

Had the Examiner provided a proper rejection, *Mateosian* nevertheless does not disclose a method including creating a first task control block for the task; creating a second task control block for the task, the second task control block having a location in a memory; and loading an address for the location of the second task control block into the first task control block. As discussed above regarding claim 1, *Mateosian* does not disclose task control block not does it disclose a task control block having a location in memory space. Furthermore, because *Mateosian* does not disclose a second task control block having a location in a memory space, it also does not disclose an address for the location of the second task control block. Thus, *Mateosian* does not

disclose the step of loading an address for the location of the second task control block into the first task control block.

Accordingly appellants submit that *Mateosian* does not disclose each and every element of claim 14. Therefore, *Mateosian* does not anticipate the present invention as recited by claim 14. For the reasons discussed in this section (Section 8.I.D), appellants respectfully request that the Board overturn the Examiner's rejection of claim 14 and all the rejected claims dependent therefrom (claims 15-16).

E. The Cited Reference Does Not Disclose a Pointer Data Structure as Recited in Claims 17 and 18

Claim 17 of the present invention recites a method, comprising:

receiving a context switching event;

saving task information for a first task in a system task control block associated with the first task, the task information for the first task including a pointer to a user task control block associated with the first task;

loading task information for a second task from a system task control block associated with the second task, the task information for the second task including a pointer to a user task control block associated with the second task.

Claim 18 of the present invention recites a method, comprising:

receiving an interrupt request;

saving a first number of state information values from a current task data structure for a currently executing task, the

> current task data structure including a pointer data structure holding a pointer to a second number of stat information values;
>
> loading a pointer to a task control block associated with an interrupt service routine into the pointer data structure;
>
> executing the interrupt service routine.

The Examiner rejected claims 17 and 18 in the same manner as the rejection against claim 14. However, aside from a brief mention of a context switching event, without discussing the way in which it is applicable to the §102 rejection, the Examiner did not discuss any of the other elements recited in claims 17 and 18. (See Final Office Action, p. 7). The Examiner failed to meet the requirements for a §102 rejection and demonstrate that each and every element of both claims 17 and 18 were disclosed by *Mateosian*. Therefore, appellants respectfully submit that the Examiner's rejection of claims 17 and 18 were improper.

Furthermore, as discussed above, *Mateosian* does not disclose task control blocks. In addition, *Mateosian* is also silent as to the task information including a pointer to a task control block.

9. Conclusions

For the reasons set forth above, appellants respectfully request that the Board reverse the final rejections of the claims by the Examiner under 35 U.S.C. §102(b), and indicate that claims 1-21 are allowable.

Respectfully submitted,

FAY, KAPLUN & MARCIN, LLP

Date: 9/7/04

By: ______________________

Michael J. Marcin
(Reg. No. 48,198)

150 Broadway, Suite 702
New York, NY 10038
(212) 619-6000
(212) 619-0276

APPENDIX A – APPEALED CLAIMS

1. A system comprising:

a memory space;

a task;

a first task control block associated with the task and located in a first area of the memory space, the first task control block including a number of first task information data structures that contain first task information;

a second task control block associated with the task and located in a second area of the memory space, the second task control block including a number of second task information data structures that contain second task information;

wherein the first area of the memory space is not directly accessible by the task, and the second area of the memory space is directly accessible by the task.

2. The system of claim 1, further comprising a current task data structure, and wherein the current task data structure is loaded with information from at least one of the first task information data structures during a context switch to execute the task.

3. The system of claim 2, further comprising a context switch routine, the context switch routine performing the context switch to execute the task.

4. The system of claim 1, wherein the first task control block includes a pointer data

structure that contains a pointer to a location of the second task control block.

5. The system of claim 1, wherein the first area of the memory space is a system space and the second area of the memory space is a user space.

6. The system of claim 1, wherein the number of second task information data structures includes error status information for the task.

7. The system of claim 1, wherein the number of second task information data structures includes a set of pointers to a set of standard modules for the task.

8. The system of claim 1, wherein the number of second task information data structures includes a pointer to environment variables for the task.

9. The system of claim 1, wherein the number of second task information data structures includes a pointer to context information for remote procedure calls made by the task.

10. The system of claim 1, wherein the number of second task information data structures includes a pointer to context information for remote procedure calls made by the task.

11. The system of claim 1, wherein the number of second task information data structures includes a pointer to exception information.

12. The system of claim 1, wherein the number of second task information data structures includes a user-definable spare field.

13. The system of claim 1, further comprising a real-time operating system.

14. A method comprising:

receiving a request to create a task;

assigning task information for the task;

creating a first task control block for the task;

loading the task information for the task into the first task control block;

creating a second task control block for the task, the second task control block having a location in a memory space;

loading an address for the location of the second task control block into the first task control block.

15. The method of claim 14, wherein the first task control block is located in a system space and the second task control block is located in a user space, the system space and

user space both within the memory space.

16. The method of claim 14, wherein the first task control block and the second task control block are located in a system space, the system space within the memory space.

17. A method, comprising:

receiving a context switching event;

saving task information for a first task in a system task control block associated with the first task, the task information for the first task including a pointer to a user task control block associated with the first task;

loading task information for a second task from a system task control block associated with the second task, the task information for the second task including a pointer to a user task control block associated with the second task.

18. A method, comprising:

receiving an interrupt request;

saving a first number of state information values from a current task data structure for a currently executing task, the current task data structure including a pointer data structure holding a pointer to a second number of stat information values;

loading a pointer to a task control block associated with an interrupt service routine into the pointer data structure;

executing the interrupt service routine.

19. The method of claim 18, further comprising:

creating the task control block associated with the interrupt service routine.

20. The method of claim 18, wherein the first number of state information values are saved on an interrupt stack.

21. The method of claim 20, wherein the second number of state information values are not saved on the interrupt stack.

APPENDIX B –

DEFINITION OF "CONTROL BLOCK" FROM

McGRAW-HILL DICTIONARY OF SCIENTIFIC

AND TECHNICAL TERMS, 4TH ED.

joint. 2. Retarded relaxation of muscle, as when it is injected with veratrine. { kən'trak·chər }

contracurrent system *See* katoptric system. { ¦kän·trə¦kər·ənt ,sis·təm }

contraflexure point [CIV ENG] The point in a structure where bending occurs in opposite directions. { ¦kän·trə'flek·shər ,point }

contrail *See* condensation trail. { 'kän,trāl }

contrail-formation graph [METEOROL] A graph containing the parameters pressure, temperature, and relative humidity for critical values at which condensation trails (contrails) form; used as an aid in forecasting the formation of condensation trails. { 'kän,trāl fȯr'mā·shən ,graf }

contraindication [MED] A symptom, indication, or condition in which a remedy or a method of treatment is inadvisable or improper. { ¦kän·trə,in·də'kā·shən }

contralateral [PHYSIO] Opposite; acting in unison with a similar part on the opposite side of the body. { ¦kän·trə'lad·ə·rəl }

CONTRAN [COMPUT SCI] Computer programming language in which instructions are written at the compiler level, thereby eliminating the need for translation by a compiling routine. { 'kän,tran }

contraorbit missile [AERO ENG] A missile that is sent backward along the calculated orbit of an aerospace weapon, satellite, or spacecraft for the purpose of destroying it in a head-on collision with an explosive warhead or through use of a secondary missile. { ¦kän·trə¦ȯr·bət 'mis·əl }

contrapositive [MATH] The contrapositive of the statement "if *p*, then *q*" is the equivalent statement "*if* not *q*, then not *p*." { ¦kän·trə'päz·əd·iv }

contrapropagating ultrasonic flowmeter [ENG] An instrument for determining the velocity of a fluid flow from the difference between the times required for high-frequency sound to travel between two transducers in opposite directions along a path having a component parallel to the flow. { ¦kän·trə'prä·pə,gād·iŋ ¦əl·trə,sän·ik 'flō,mēd·ər }

contrarotating propellers [MECH ENG] A pair of propellers on concentric shafts, turning in opposite directions. { ¦kän·trə'rō,tād·iŋ prə'pel·ərz }

contrarotation [ENG] Rotation in the direction opposite to another rotation. { ¦kän·trə·rō'tā·shən }

contra solem [METEOROL] Characterizing air motion that is counterclockwise in the Northern Hemisphere and clockwise in the Southern Hemisphere; literally, against the sun. { 'kän·trə 'sō,lem }

contrast [COMMUN] The degree of difference in tone between the lightest and darkest areas in a television or facsimile picture. [COMPUT SCI] In optical character recognition, the difference in color, reflectance, or shading between two areas of a surface, for example, a character and its background. { 'kän,trast }

contrast control [ELECTR] A manual control that adjusts the range of brightness between highlights and shadows on the reproduced image in a television receiver. { 'kän,trast kən'trōl }

contrastes [METEOROL] Winds a short distance apart blowing from opposite quadrants, frequent in the spring and fall in the western Mediterranean. { kōn'tras,tēz }

contrast ratio [ELECTR] The ratio of the maximum to the minimum luminance values in a television picture. { 'kän,trast ,rā·shō }

contrast sensitivity *See* threshold contrast. { 'kän,trast sen·sə'tiv·əd·ē }

contrast sharpening [GRAPHICS] A procedure for increasing local contrasts in an image and reducing blurring by using computer processing of the image to emphasize its high spatial frequencies. { 'kän,trast ,shär·pən·iŋ }

contrast threshold *See* threshold contrast. { 'kän,trast ,thresh,hōld }

contravariant functor [MATH] A functor which reverses the sense of morphisms. { ¦kän·trə'ver·ē·ənt 'fəŋk·tər }

contravariant index [MATH] A tensor index such that, under a transformation of coordinates, the procedure for obtaining a component of the transformed tensor for which this index has the value *p* involves taking a sum over *q* of the product of a component of the original tensor for which the index has the value *q* times the partial derivative of the *p*th transformed coordinate with respect to the *q*th original coordinate; it is written as a superscript. { ¦kän·trə'ver·ē·ənt 'in,deks }

contravariant tensor [MATH] A tensor with only contravariant indices. { ¦kän·trə'ver·ē·ənt 'ten·sər }

contravariant vector [MATH] A contravariant tensor of degree 1, such as the tensor whose components are differentials of the coordinates. { ¦kän·trə'ver·ē·ənt 'vek·tər }

contributory *See* tributary. { kən'trib·yə,tȯr·ē }

control [COMPUT SCI] **1.** The section of a digital computer that carries out instructions in proper sequence, interprets each coded instruction, and applies the proper signals to the arithmetic unit and other parts in accordance with this interpretation. **2.** A mathematical check used with some computer operations. [CONT SYS] A means or device to direct and regulate a process or sequence of events. [ELECTR] An input element of a cryotron. [STAT] **1.** A test made to determine the extent of error in experimental observations or measurements. **2.** A procedure carried out to give a standard of comparison in an experiment. **3.** Observations made on subjects which have not undergone treatment, to use in comparison with observations made on subjects which have undergone treatment. { kən'trōl }

control accuracy [CONT SYS] The degree of correspondence between the ultimately controlled variable and the ideal value in a feedback control system. { kən'trōl ,ak·yə·rə·sē }

control agent [CHEM ENG] In process automatic-control work, material or energy within a process system of which the manipulated (controlled) variable is a condition or characteristic. { kən'trōl ,ā·jənt }

control and read-only memory [COMPUT SCI] A read-only memory that also provides storage, sequencing, execution, and translation logic for various microinstructions. Abbreviated CROM. { kən'trōl ən ¦rēd ,ōn·lē 'mem·rē }

control area [NAV] An area over which air-traffic control is exercised by a ground-located traffic-control center. { kən'trōl ,er·ē·ə }

control bit [COMPUT SCI] A bit which marks either the beginning or the end of a character transmitted in asynchronous communication. { kən'trōl ,bit }

control block [COMPUT SCI] A storage area containing (in condensed, formalized form) the information required for the control of a task, function, operation, or quantity of information. { kən'trōl ,bläk }

control board [ELEC] A panel at which one can make circuit changes, as in lighting a theater. [ENG] A panel in which meters and other indicating instruments display the condition of a system, and dials, switches, and other devices are used to modify circuits to control the system. Also known as control panel; panel board. { kən'trōl ,bȯrd }

control break [COMPUT SCI] A key change which takes place in a control data field, especially in the execution of a report program. { kən'trōl ,brāk }

control card [COMPUT SCI] A punched card containing input data or parameters which are necessary to begin or modify a program, or containing instructions needed for the specific application of a general routine. { kən'trōl ,kärd }

control change [COMPUT SCI] A change of function that occurs when successive records (such as those entered on punched cards) differ in the data entered in the control field; for example, a punched-card tabulator may change from adding to printing at the end of a series of items. Also known as comparing control change. { kən'trōl ,chānj }

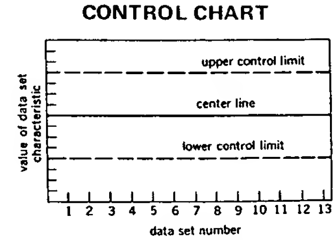
control character [COMPUT SCI] A character whose occurrence in a particular context initiates, modifies, or stops a control operation in a computer or associated equipment. { kən'trōl ,kar·ik·tər }

control characteristic [ELECTR] **1.** The relation, usually shown by a graph, between critical grid voltage and anode voltage of a gas tube. **2.** The relation between control ampere-turns and output current of a magnetic amplifier. { kən'trōl ,kar·ik·tə'ris·tik }

control chart [IND ENG] A chart in which quantities of data concerning some property of a product or process are plotted and used specifically to determine the variation in the process. { kən'trōl ,chärt }

control circuit [COMPUT SCI] One of the circuits that responds to the instructions in the program for a digital computer. [ELEC] A circuit that controls some function of a machine, device, or piece of equipment. [ELECTR] The circuit that feeds the control winding of a magnetic amplifier. { kən'trōl ,sər·kət }

CONTROL CHART



Control chart, consisting of a center line and two sets of control lines.